AMENDMENTS TO THE CLAIMS

Below is the entire set of pending claims pursuant to 37 C.F.R §1.121(c)(3)(i), with any mark-ups showing the changes made by the present Amendment.

1-2 (Canceled)

3. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~23, wherein:

said shift register shifts ~~(40) the~~ said configuration data modules among said ~~plurality of data storage units (41a to 41d)~~ data registers circularly; and

said logic block refers only to said one configuration data module stored in said one data register.

4. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~25, ~~comprising~~ wherein:

~~a~~ said selector ~~(42) which selects at least one of said plurality of data storage units (49a to 49d),~~ changes selection of said one data memory with another data memory among said data memories circularly; and

~~wherein~~ said logic ~~computing unit (43)~~ block refers to ~~the~~ said one configuration data module ~~which is~~ stored in said ~~data storage unit selected by said selector (42)~~ one data memory.

5. (Canceled)

6. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~ 23, ~~comprising:~~

~~a parameter register (45) which stores all or part of internal parameters of said logic computing unit (43) for stacking;~~

~~a detector (44) which detects a call and a call back by one of the plurality of configuration data modules to another one of the plurality of configuration data modules; and~~

~~a controller (47) which controls logic computing by said logic computing unit,~~ wherein said controller ~~(47):~~

~~stores the internal parameters of said logic computing unit in said parameter register (45), when said detector (44) detects a call by one of the plurality of configuration data modules to another one of the plurality of configuration data modules as a subroutine; and~~

restores the ~~internal~~ parameters stored in said parameter ~~register (45) in~~ buffer to said ~~logic computing unit (43), when said detector (44) detects a call back to one of the plurality of configuration data modules~~ flip flop(s) when one configuration data module stored in said one data register calls back to another configuration data module as a main routine.

7. (Canceled)

8. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~ 23, ~~comprising~~ wherein:

~~a detector (44) which~~ said routine detector detects a call by ~~one of the plurality of~~ said one configuration data ~~modules~~ module to said another ~~one of the plurality of~~ configuration data ~~modules; and~~ module as the subroutine;

~~a controller (47) which controls logic computing by said logic computing unit (43), wherein~~ said controller ~~(47):~~

~~searches, when said detector (44) detects a call by one of the plurality of configuration data modules to another one of the plurality of configuration data modules as a subroutine, said plurality of data storage units (41a to 41d, 49a to 49d) for the configuration data module as the subroutine; and~~

~~sends a load command to said loader (3), in a case where the configuration data modules as the subroutine is not searched out, and said loader (3) loads the configuration data modules as the subroutine which is indicated by the load command to one of said plurality of data storage units (41a to 41d, 49a to 49d)~~

searches said data registers for said another configuration data module as the subroutine when said routine detector detects the call, and sends a load command to said loader in a case where said another configuration data module as the subroutine is not searched out; and

said loader loads said another configuration data module as the subroutine from said data storage to one of said data registers in response to the load command received from said controller.

9. (Canceled)

10. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~ 23, comprising a compiler ~~(6)~~ which creates each of ~~the plurality of~~ said configuration data modules based on each of a plurality of source program modules.

11-12. (Canceled)

13. (Currently amended) The ~~logic~~ computing method according to claim ~~12~~ 30, comprising:

____shifting, by said shift register, ~~(40) the~~ said configuration data modules among said ~~plurality of data storage units (41a to 41d)~~ data registers circularly; and

referring, by said logic block, only to said one configuration data module stored in said one data register.

14. (Currently amended) The ~~logic~~ computing method according to claim ~~11~~ 32, comprising:

~~selecting~~ changing selection of said one data memory, by a selector ~~(42)~~, ~~at least one of said plurality of data storage units (49a to 49d)~~ with another data memory among said data memories circularly; and

referring, by said logic block, to ~~the~~ said one configuration data module stored in said ~~data storage unit selected by said selector (42)~~ one data memory.

15. (Canceled)

16. (Currently amended) The ~~logic~~ computing method according to claim ~~11~~ 30, comprising:

~~detecting a call by one of the plurality of configuration data modules to another one of the plurality of configuration data modules as a subroutine;~~

~~storing all or part of internal parameters of said logic computing unit (43) in a parameter register (45) in response to said detecting; and~~

restoring, by said controller, the ~~internal~~ parameters stored in said parameter ~~register (45) in~~ buffer to said ~~logic computing unit (43), when one of the plurality of configuration~~

~~data modules is called back~~ flip flop(s) when one configuration data module stored in said one data register calls back to another configuration data module as a main routine.

17. (Canceled)

18. (Currently amended) The ~~logic~~ computing method according to claim ~~17~~ 30, comprising:

detecting, by said routine detector, a call by said one configuration data module to said another configuration data module as the subroutine;

searching, ~~when a call by one of the plurality of configuration data modules to another one of the plurality of configuration data modules as a subroutine is detected, said plurality of data storage units (41a to 41d, 49a to 49d) for the configuration data module as the subroutine~~ by said controller, said data registers for said another configuration data module as the subroutine when the call is detected by said routine detector;

sending, by said controller, a load command to said loader ~~(3),~~ in a case where ~~the~~ said another configuration data module as the subroutine is not searched out; and

loading, by said loader, ~~(3) the~~ said another configuration data module as the subroutine ~~which is indicated by the load command to one of said plurality of data storage units (41a to 41d, 49a to 49d)~~ from said data storage to one of said data registers in response to the load command received from said controller.

19. (Canceled)

20. (Currently amended) The ~~logic~~ computing method according to claim ~~11~~ 30, comprising creating, by a compiler ~~(6),~~ each of ~~the plurality of~~ said configuration data

modules based on each of a plurality of source program modules.

21. (Currently amended) The ~~logic~~ computing system according to claim ~~1~~ 23, ~~further comprising a loader (3) which loads the configuration data module(s) from outside the logic computing unit to one or more of said plurality of data storage units (41a to 41d, 49a to 49d),~~ wherein:

said logic computing ~~unit (43)~~ device generates a load command in the process of computing by said ~~plurality of programmable logic circuits (43a)~~ logic block; and

said loader ~~(3)~~ loads the configuration data module, ~~which is~~ indicated by the load command in the process, from said data storage to one of said data registers.

22. (Currently amended) The ~~logic~~ computing method according to claim ~~11~~ 30, further comprising:

generating a load command in the process of computing by said ~~plurality of programmable logic circuits (43a)~~ logic block; and

loading, by ~~a~~ said loader ~~(3)~~, the configuration data module, ~~which is~~ indicated by the load command in the process, from ~~outside the logic computing unit~~ said data storage to one of said ~~plurality of data storage units (41a to 41d, 49a to 49d)~~ one of said data registers.

23. (New) A computing system comprising:

a data storage;

a loader; and

a logic computing device,

wherein said data storage stores a plurality of configuration data modules each of which includes data for forming a look-up table, said loader loads said configuration data modules from said data storage to said logic computing device in response to a plurality of load commands,

said logic computing device comprises a shift register, a logic block, a routine detector, a parameter register, a parameter buffer and a controller,

wherein said shift register includes a plurality of data registers each of which stores one of said configuration data modules loaded by said loader,

said logic block comprises at least one gate circuit(s) and at least one flip flop(s), and provides a logical function value of logic input data to outside said logic computing device as logic output data,

said routine detector identifies a relationship between one configuration data module stored in one data register of said data registers and another configuration data module,

said parameter register stores at least part of parameters stored in said flip flop(s) when said configuration data modules stored in said shift register are shifted,

said parameter buffer stores at least part of parameters stored in said flip flop(s) when said one configuration data module stored in said one data register calls another configuration data module as a subroutine, and

said controller controls logic computing in said logic block.

24. (New) The computing system according to claim 23, wherein said controller returns the parameters stored in said parameter buffer to said logic block as input of said gate circuit(s) when said another configuration data module as the subroutine is shifted to said one data register.

25. (New) A computing system comprising:

a data storage;

a loader; and

a logic computing device,

wherein said data storage stores a plurality of configuration data modules each of which includes data for forming a look-up table,

said loader loads said configuration data modules from said data storage to said logic computing device in response to a plurality of load commands received from said logic computing device,

said logic computing device comprises a selector, a plurality of data memories, a logic block, a routine detector, a parameter register, a parameter buffer and a controller,

wherein each of said data memories stores one of said configuration data modules loaded by said loader,

said selector selects one data memory of said data memories in accordance with a control signal received from said controller,

said logic block comprises at least one gate circuit(s) and at least one flip flop(s), and provides a logical function value of logic input data to outside said logic computing device as logic output data,

said routine detector identifies a relationship between one configuration data module stored in said one data memory and another configuration data module,

said parameter register stores at least part of parameters stored in said flip flop(s) when said selector changes selection of said one data memory,

said parameter buffer stores at least part of parameters stored in said flip flop(s) when said one configuration data module stored in said one data memory calls another configuration data module as a subroutine, and

said controller controls logic computing in said logic block.

26. (New) The computing system according to claim 25, wherein said controller restores the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data memory calls back to another configuration data module as a main routine.

27. (New) The computing system according to claim 25, wherein:

said routine detector detects a call by said one configuration data module to said another configuration data module as the subroutine;

said controller searches said data memories for said another configuration data module as the subroutine when said routine detector detects the call, and sends a load command to said loader in a case where said another configuration data module as the subroutine is not searched out; and

said loader loads said another configuration data module as the subroutine from said data storage to one of said data registers in response to the load command received from said controller.

28. (New) The computing system according to claim 25, further comprising a compiler which creates each of said configuration data modules based on each of a plurality of source program modules.

29. (New) The computing system according to claim 25, wherein:

said logic computing device generates a load command in the process of computing by said logic block; and

said loader loads the configuration data module, indicated by the load command in the process, from said data storage to one of said data memories.

30. (New) A computing method comprising:

storing a plurality of configuration data modules in a data storage, each of said configuration data modules includes data for forming a look-up table;

loading, by a loader, said configuration data modules from said data storage to a logic computing device in response to a plurality of load commands;

storing each of said configuration data modules loaded by said loader in each of a plurality of data registers included in a shift register of said logic computing device;

configuring a logic block of said logic computing device so as to comprise at least one gate circuit(s) and at least one flip flop(s);

providing, by said logic block, a logical function value of logic input data to outside said logic computing device as logic output data;

identifying, by a routine detector of said logic computing device, a relationship between one configuration data module stored in one data register of said data registers and another configuration data module;

storing at least part of parameters, stored in said flip flop(s), in a parameter register of said logic computing device when said configuration data modules stored in said shift register are shifted;

storing at least part of parameters, stored in said flip flop(s), in a parameter buffer of said logic computing device when said one configuration data module stored in said one data register calls another configuration data module as a subroutine; and

controlling, by a controller of said logic computing device, logic computing in said logic block.

31. (New) The computing method according to claim 30, further comprising returning, by said controller, the parameters stored in said parameter buffer to said logic block as input of said gate circuit(s) when said another configuration data module as the subroutine is shifted to said one data register.

32. (New) A computing method comprising:

storing a plurality of configuration data modules in a data storage, each of said configuration data modules includes data for forming a look-up table;

loading, by a loader, said configuration data modules from said data storage to a logic computing device in response to a plurality of load commands received from said logic computing device;

storing each of said configuration data modules loaded by said loader in each of a plurality of data memories of said logic computing device;

selecting, by a selector of said logic computing device, one data memory of said data memories in accordance with a control signal received from a controller of said logic computing device;

configuring a logic block of said logic computing device so as to comprise at least one gate circuit(s) and at least one flip flop(s);

providing, by said logic block, a logical function value of logic input data to outside said logic computing device as logic output data;

identifying, by a routine detector of said logic computing device, a relationship between one configuration data module stored in said one data memory and another configuration data module;

storing at least part of parameters, stored in said flip flop(s), in a parameter register of said logic computing device when said selector changes selection of said one data memory;

storing at least part of parameters, stored in said flip flop(s), in a parameter buffer of said logic computing device when said one configuration data module stored in said one data memory calls another configuration data module as a subroutine; and

controlling, by said controller, logic computing in said logic block.

33. (New) The computing method according to claim 32, further comprising restoring, by said controller, the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data memory calls back to another configuration data module as a main routine.

34. (New) The computing method according to claim 32, further comprising:

detecting, by said routine detector, a call by said one configuration data module to said another configuration data module as the subroutine;

searching, by said controller, said data memories for said another configuration data module as the subroutine when the call is detected by said routine detector;

sending, by said controller, a load command to said loader when said another configuration data module as the subroutine is not searched out; and

loading, by said loader, said another configuration data module as the subroutine from said data storage to one of said data memories in response to the load command received from said controller.

35. (New) The computing method according to claim 32, further comprising creating, by a compiler, each of said configuration data modules based on each of a plurality of source program modules.

36. (New) The computing method according to claim 32, further comprising:

generating a load command in the process of computing by said logic block; and

loading, by said loader, the configuration data module, indicated by the load command in the process, from said data storage to one of said data memories.